PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 9/00, 13/14 A1

(11) International Publication Number: WO 94/14114

(43) International Publication Date: 23 June 1994 (23.06.94)

(21) International Application Number: PCT/US93/11506

(22) International Filing Date: 29 November 1993 (29.11.93)

(30) Priority Data:
987,365 7 December 1992 (07.12.92) US

(71) Applicant: OVERLORD, INC. [US/US]; 90 South LaSalle Street, Suite 2760, Chicago, IL 60603 (US).

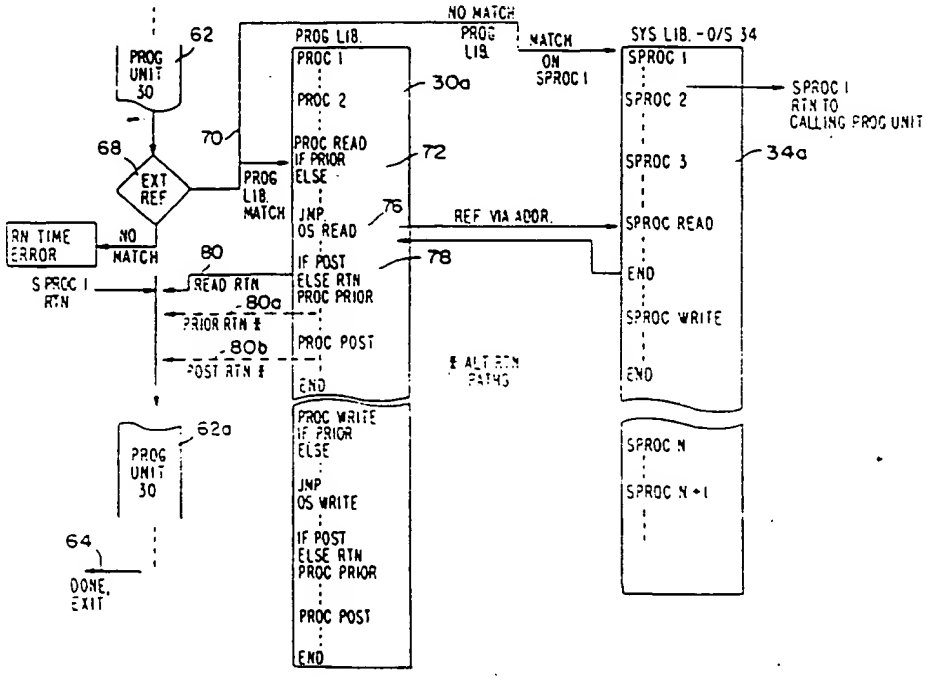
(72) Inventor: KENNEDY, Donald, J.; 1725 Westbridge Court, Southbridge Commons, Schamburg, IL 60194 (US).

(74) Agents: VARGO, Paul, M. et al.; Dressler, Goldsmith, Shore & Milnamow, Two Prudential Plaza, Suite 4700, Chicago, IL 60601 (US).

(81) Designated States: AU, FL HU, JP, KR, NO, NZ, PL, RU, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: INTERCEPTION SYSTEM AND METHOD INCLUDING USER INTERFACE



(57) Abstract

A method of intercepting pre-existing computer instructions in order to modify and/or enhance pre-existing program units (30) and supply user entry points determines, in one or more embodiments, if a reference can be found in a program unit (30). If so located, the corresponding method provides user code entry points (steps 72, 78) before and after the intercepted instruction, perhaps in modified and/or enhanced form, is executed (step 76). Blocks of user supplied code can be provided at the entry points to enhance, upgrade, and/or expand upon the intercepted instruction, thereby enhancing the pre-existing program unit (30).

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgystan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SI	Slovenia
CI	Côte d'Ivoire	KZ	Kazakhstan	SK	Slovakia
CM	Cameroon	LI	Liechtenstein	SN	Senegal
CN	China	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
CZ	Czech Republic	LV	Latvia	TJ	Tajikistan
DE	Germany	MC	Monaco	TT	Trinidad and Tobago
DK	Denmark	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	US	United States of America
FI	Finland	ML	Mali	UZ	Uzbekistan
FR	France	MN	Mongolia	VN	Viet Nam
GA	Gabon				

INTERNATIONAL SEARCH REPORT

International application No
PCT/US93/11506

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :G06F 9/00; 13/14

US CL :395/375

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/375, 700

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS: search terms: service routine, request, call

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US, A, 5,109,515 (Laggis et al). 28 April 1992 See abstract. See column 2, lines 37-60, column 3, lines 4-29, column 5, lines 44-68, column 6, lines 14-36, column 7, lines 11-35, and figures 3 and 4.	1-9
Y,P	US, A, 5,241,634 (Suzuki) 31 August 1993 See abstract, column 3, lines 37-68, column 4, lines 1-6 and 11-47.	1-9
X Y	US, A, 4,768,150 (Chang et al) 30 August 1988 See abstract, column 2, lines 65-68, and column 3, lines 1-53.	1-9

[X] Further documents are listed in the continuation of Box C [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 19 January 1994	Date of mailing of the international search report MAR 03 1994
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No NOT APPLICABLE	Authorized officer Parsh Lall Telephone No (703) 305-9715

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No
PCT/US93/11506

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No
X --- Y	US, A, 5,124,909 (Blakely et al) 23 June 1992 See abstract, column 1, lines 25-68, and column 2, lines 1-13.	1-9

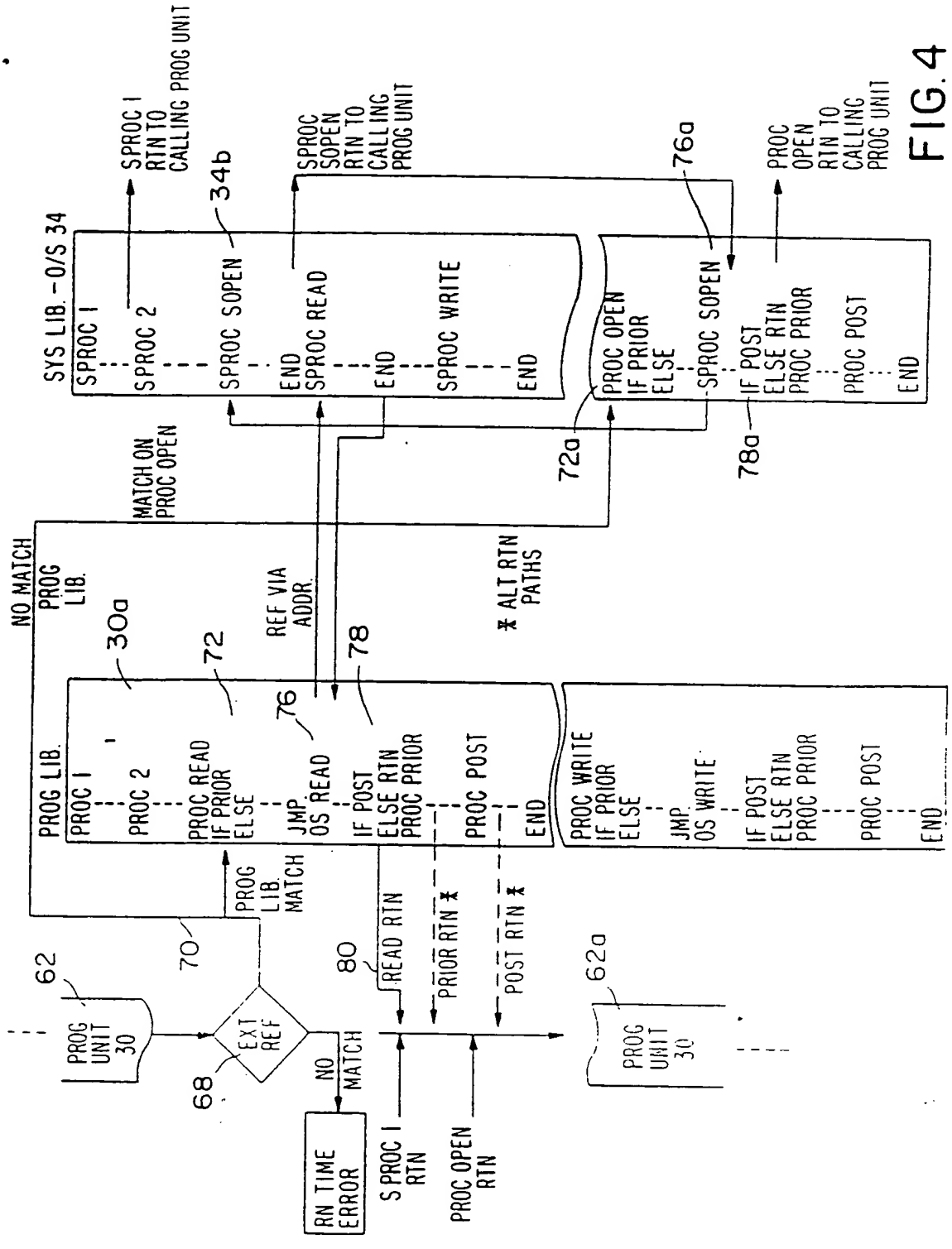

SYS LIB. - O/S 34
SPROC 1
SPROC 2
SPROC SOPEN
END
SPROC READ
END
SPROC WRITE
END
SPROC 1 RTN TO CALLING PROG UNIT
SPROC SOPEN RTN TO CALLING PROG UNIT
PROC OPEN RTN TO CALLING PROC UNIT
34b
76a
72a
78a
PROC OPEN
IF PRIOR
ELSE
SPROC SOPEN
IF POST
ELSE RTN
PROC PRIOR
PROC POST
END
MATCH ON PROC OPEN
NO MATCH PROG LIB.
30a
72
REF VIA ADDR.
78
76
* ALT RTN PATHS
PROG LIB.
PROC 1
PROC 2
PROC READ
IF PRIOR
ELSE
JMP.
OS READ
IF POST
ELSE RTN
PROC PRIOR
PROC POST
END
PROC WRITE
IF PRIOR
ELSE
JMP.
OS WRITE
IF POST
ELSE RTN
PROC PRIOR
PROC POST
END
PROG LIB. MATCH
70
62
PROG UNIT 30
EXT REF
68
NO MATCH
RN TIME ERROR
80
READ RTN
PRIOR RTN *
POST RTN *
S PROC 1 RTN
PROC OPEN RTN
62a
PROG UNIT 30

FIG. 4

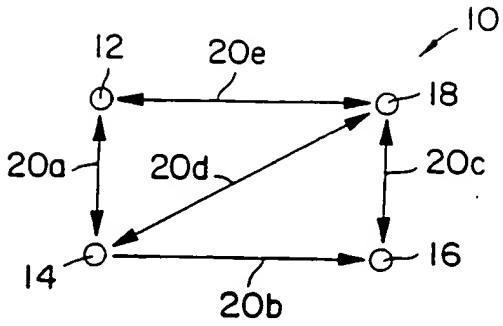

10
12
20e
18
20a
20d
20c
14
16
20b

FIG. 1

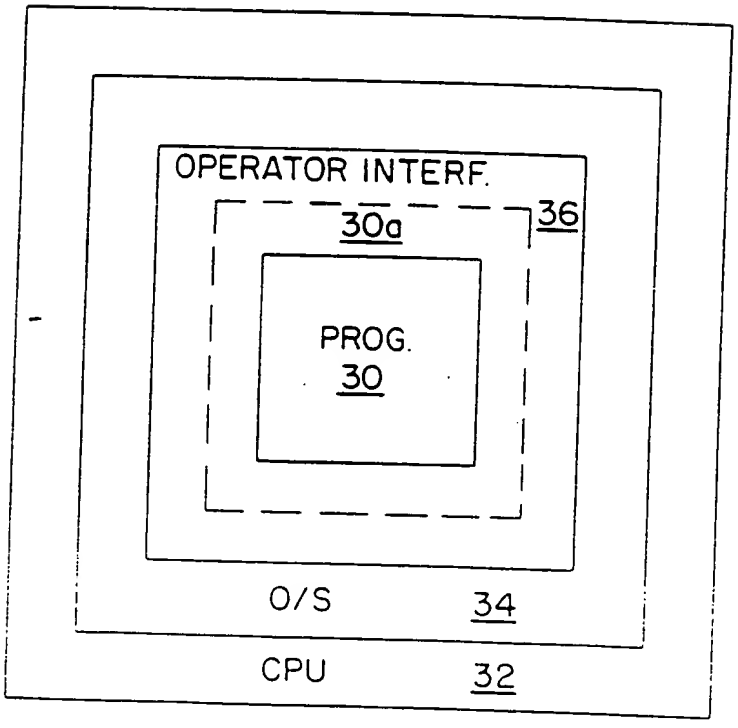

OPERATOR INTERF.
30a
36
PROG.
30
O/S
34
CPU
32

FIG. 2

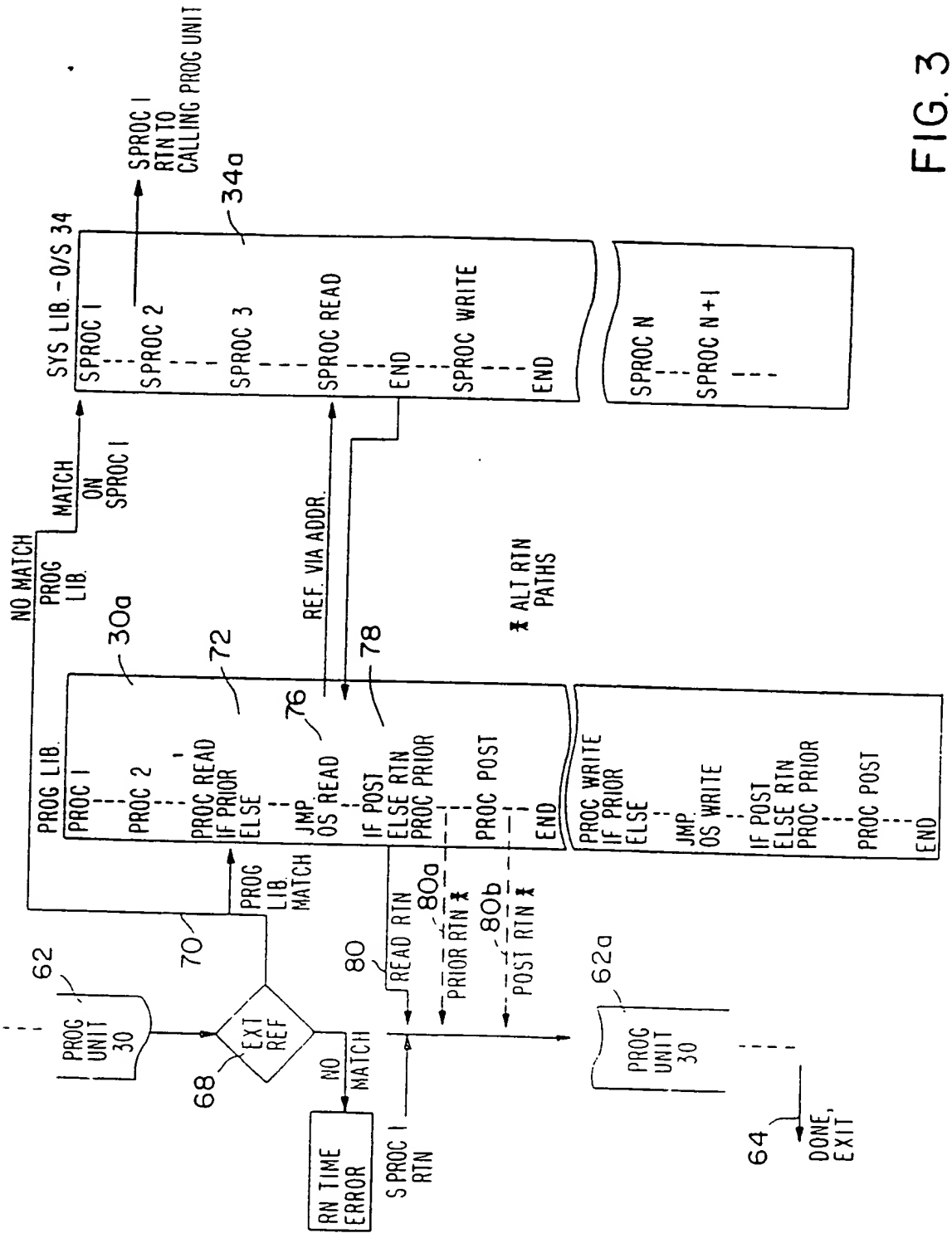

PROG UNIT 30
62
EXT REF
68
NO MATCH
RN TIME ERROR
PROG LIB. MATCH
70
NO MATCH
PROG LIB.
MATCH ON SPROC 1
SYS LIB. - O/S 34
SPROC 1
SPROC 2
SPROC 3
SPROC READ
END
SPROC WRITE
END
SPROC N
SPROC N+1
SPROC 1 RTN TO CALLING PROG UNIT
34a
30a
72
REF. VIA ADDR.
78
76
PROG LIB.
PROC 1
PROC 2
PROC READ IF PRIOR ELSE
JMP. OS READ
IF POST ELSE RTN PROC PRIOR
PROC POST
END
PROC WRITE IF PRIOR ELSE
JMP. OS WRITE
IF POST ELSE RTN PROC PRIOR
PROC POST
END
* ALT RTN PATHS
80
READ RTN
80a
PRIOR RTN *
80b
POST RTN *
S PROC 1 RTN
62a
PROG UNIT 30
64
DONE, EXIT

FIG. 3

INTERCEPTION SYSTEM AND METHOD INCLUDING USER INTERFACE

Field of the Invention

The invention relates to single and multiprocessor computer systems that supply system services to requesting program units running on or in such systems. More particularly, the invention relates to methods of enhancing or modifying the run-time operation of selected, pre-existing program units.

Background of the Invention

Computer systems have, over a period of years, evolved from stand-alone individual processors to various forms of multi-processor systems. Many computer systems use program units, sometimes referred to simply as "programs".

The program units contain computer instructions which the computer system can execute in order to perform specific functions. These program units may have been created from other program units. However, in most cases, a human being was involved at some point in the creation of the set of computer instructions being executed.

Program units are intended to meet certain known or projected needs when implemented. However, most program units designed in the past or being designed in the present will not conform to all future needs.

Prior art systems have approached the need to be flexible to deal with future needs in many ways. In many cases, prior approaches have not been cost effective and/or do not allow the user many options on their implementation.

The evolution and combination of new hardware systems, new operating systems, new program units, new system procedures, new data structures, or new user interfaces may require that the original program units be modified, recompiled, or worse, abandoned due to compatibility and/or cost related problems. Some of the prior art approaches require extensive training on both the use and implementation of these methods. Some users may not be able to afford the time, money, and human resources to implement the prior art approaches.

This need for flexibility in updating or modifying existing programs is especially apparent in multi-processor distributed systems. Several different types of problems have provided the impetus to the drive toward multi-processor systems.

One impetus has been a desire to share information more effectively among diverse users. An approach to this problem has been to couple a variety of processors, which may or may not be the same, together via a local area network. Such networks enable many different individuals and their associated processors to have access to common information and to have access to one another.

Yet another impetus toward multi-processor environments has been a desire to create highly reliable computer systems out of less reliable components. Such systems are typically used in environments such as banking, transaction processing, or inventory control, wherein reliability is of paramount importance.

One such family of computer systems is marketed by Tandem of Cupertino, California. Tandem systems can be implemented in stand-alone, multiple processor configurations, or as multiple interconnected nodes. Each node corresponds to one or more multiple processor systems.

Where major program systems, which might include dozens of program units, to support multiple remote transaction terminals or inventory control functions are installed and running on a production basis in a multiple processor environment, the above-noted problem of updating and maintaining program units becomes very difficult and expensive to solve. For example, a new operating system might be adopted by the hardware vendor. In such an instance, the system operator might have to install the new operating system to receive continuing support and operating system maintenance.

If the change in operating systems is not transparent to the existing program systems, they may need to be modified or recompiled. This process is not only expensive and time consuming, but in a multi-program, multi-processor environment can result in errors which could cause catastrophic results.

In addition, where the software had been obtained from a third party vendor, the user might not have the source code or documentation necessary to make modifications, expansions, or recompilations. Worse yet, the third party vendor will, in all likelihood, not continue to support or provide new releases to the user.

Thus, there continues to be a need to be able to safely upgrade or modify existing programs in a cost effective fashion as the requirements or the environment change. Preferably, this need could be met by system operating personnel without a need to return to the original software vendor or to modify the original provided program units.

In addition, in a multiple processor system, the operating environment is continuously changing. As a result, the mix of resources, available processors, and the like, available each time a program unit or a

process is initiated, will be different, depending on what other program units or processes are active at any given time.

Thus, there is continuous problem of resource allocation and management which must be addressed in such systems. One known approach, marketed by the assignee of the present application under the name of "Automatic Network Balancing System" for Tandem computers, provides resource allocation services and resource management in such environments based on predetermined and fixed allocation methods.

In the known automatic network balancing system, the performance factors which are taken into account to select the best or most appropriate processor to which a process is to be allocated, include availability or busy state of a given processor, available memory, swap rate, dispatch rate, memory queue length, jobs that are available on the ready list, as well a number of others. The various performance factors are evaluated using a weighing system. The processor which appears to be most appropriate is then selected to run the process.

The known load balancing system has been very successful and can be used to substantially increase performance of Tandem-like systems. Nevertheless, the method of selecting the most appropriate processor to be allocated to carry out a given task does not take into account site or user needs for diversity or customization between one installation and another.

Thus, there continues to be a need for a more flexible approach which can take into account variations from site to site. Preferably, such an approach could be implemented to allow site specific input to the processor selection process or to expand upon the services provided to a given process which is being

executed. Preferably, the implementation will be transparent to the respective process.

Summary of the Invention

This invention is directed to an apparatus and a method of run-time interception of pre-existing computer instructions in program units in order to support user hooks or entry points which can be used to modify and/or enhance the originating and/or receiving program units, at the user's discretion. As a result, the program units can meet the user's present needs and allow modification by the users, on an as needed basis, to support the future needs. Using the present invention, this can be accomplished without requiring the support and/or guidance and/or expertise of the original authors and/or inventors of the program units being intercepted or any additional physical, electronics, or mechanical device.

The above result is achieved by intercepting system service calls which are made by executing program units at run time when the program units request that the operating system of the computer system provide a service on their behalf. The interception can take place in the main program units, user library program units, system library program units, or a combination of the program units listed above.

The method also contemplates that the interception of the system service calls and user hooks or entry points would be placed in several types of program units. This gives the users many options as to where the interceptions of the system service calls will take place. Further, it allows the user to implement the invention on a program unit by program unit basis, if desired, or to implement the invention on a system by system basis.

In accordance with one aspect of the invention, an apparatus and a method are provided for altering or translating one or more steps of a pre-existing method for carrying out a predetermined function. Site or user defined steps or functions can be incorporated into the process for customization or specialization.

The method can be used, for example, for allocating resources within a multiple processor computer system. In other aspects of the invention, different types of functions can be implemented beyond those specified in the pre-existing method.

The method includes detecting a step which is a candidate for alteration. The alteration process could include carrying out a different function from that which the step initially requested, or for translating or expanding upon the step.

A determination is made if a previously defined, user supplied, pre-alteration set of steps is to be executed before carrying out one or more predetermined altering or translating steps. In response to this determining step, the group of site or user supplied pre-alteration or pre-translation steps is executed as indicated.

The method then includes executing the one or more predefined altering or translating steps. Such steps could include, in accordance with one aspect of the invention, determining which of a plurality of available resources is to be used to carry out the requested step which is the candidate for alteration.

Alternately, the predefined altering steps could provide enhanced functions not called for in the original candidate steps. Such enhanced functions may have become desirable, so long as they can be provided

so as to be transparent to the original candidate-step or steps.

The method then makes a determination as to whether or not there are one or more post-alteration, site or user supplied steps. These steps can then be executed as indicated after executing the set of altering steps.

In accordance with yet another aspect of the invention, the method can be used for the purpose of allocating resources within a multiple node, multiple processor system. Each of the nodes can include one or more computer processors. The nodes can be physically displaced from one another, and can be coupled together via communication lines.

This aspect includes the steps of:

carrying out a sequence of steps in a predetermined process;

detecting a step in the sequence which is to be carried out and which is a candidate for translation;

intercepting the detected step and determining if a previously defined, user supplied, pre-translation set of steps exists;

interrupting the sequence and executing the user supplied pre-translation set of steps as indicated;

translating the candidate step into a predetermined sequence of one or more predetermined translated steps;

subsequent to the translation step, determining if a previously defined, user supplied, post-translation set of steps exists;

executing the user supplied, post-translation set of steps as indicated; and

returning to the sequence of steps immediately after the detected step, thereby continuing the process.

In yet another aspect of the invention, the method can be used for the purpose of resource allocation for the purpose of not only optimizing processing throughput, but also for the purpose of creating redundant databases automatically in spaced apart locations for purposes of other functions, such as disaster recovery, for instance.

These and other aspects and attributes of the present invention will be discussed subsequently with reference to the following drawings and accompanying specification.

Brief Description Of The Drawing

Figure 1 is a schematic diagram of a multiple node, multiple processor network;

Figure 2 is a schematic diagram of an environment in which a program unit might be executed;

Figure 3 is a flow diagram of a method in accordance with the present invention; and

Figure 4 is a flow diagram of an alternate method in accordance with the present invention.

Detailed Description of the Preferred Embodiment

While this invention is susceptible of embodiment in many different forms, there is shown in the drawing, and will be described herein in detail, specific embodiments thereof with the understanding that the present disclosure is to be considered as an exemplification of the principles of the invention and is not intended to limit the invention to the specific embodiments illustrated.

The present method makes it possible for a program user or a system operator to update and modify pre-existing programs without requiring the recompiling of the source codes of the respective program unit(s).

This is accomplished by intercepting selected calls or references to procedures, program units, or variables that can be external or internal to a pre-existing executing program unit. One type of interceptable instruction is an operating system service call.

On interception, the operating system will look for the called procedure in a library linked to the executing program unit, if such exists. In the absence of a program related library, or in the absence of a match with the called procedure in the executing program unit, the operating system will then attempt to find the called procedure or program unit in its system library.

Where a match is found in either the program library or the system library, that procedure or program unit is then executed. If there is no match, an indication of a run-time error should be returned to the calling program unit.

The present method makes available "user hooks" in the respective library procedures or program units. The phrase "user hooks" as used herein refers to intentionally created entry points or steps wherein a user or system operator can insert one or more computer instructions (blocks of code) for the purpose of transparently updating or modifying the executing program unit. Hence, the user has greater control over its computer system(s) and is able to make modifications or enhancements outside of the executing program unit. This avoids any need to modify or recompile that program unit.

Another advantage of the present method is that it can be used where the program library is incorporated into the program unit itself. The user hooks provide a way for a user or operator to create a bridge between various versions or releases of software packages, as well as program units.

Figure 1 illustrates schematically a multiple processor computer network 10. The network 10 includes a plurality of nodes 12 through 18.

Each of the nodes 12 through 18 can include one or more computer systems. Representative examples include Tandem-type multiple processor computer systems which might include up to 16 processor modules.

It will be understood that a node, such as node 12, could be implemented as a stand-alone, single processor computer system. Neither the number of processors, nor the architecture thereof, nor the presence or absence of communication links are a limitation of the present invention. The present invention can be advantageously practiced in conjunction with a single, stand-alone system.

Each of the nodes 12 through 18 can communicate with at least one other node via communication channels, such as the channels 20a through 20e. The network 10 can be geographically disbursed with the nodes 12 through 18 coupled, at least in part, via long distance communication links or other communications methods.

Figure 2 illustrates schematically a program unit 30 which is to be executed on a processor 32. As is conventional, the program unit 30 communicates with the processor 32 via an operating system 34. The operating system 34 provides a variety of services to the executing program unit.

The program unit 30 and operating system 34 would normally be stored in one or more storage devices or units of the processor 32. The details of such storage and the process wherein the operating system 34 initiates executing of the program unit 30 on the processor 32 are known and are not a limitation of the present invention.

As has long been recognized, one aspect of an operating system is to enhance the efficiency of utilization of the processor 32 as well as to improve the speed and ease of creation of programs such as the program unit 30. In this regard, the operating system 34 provides a variety of predefined commands, so-called "System Service Calls" (SSC), which carry out certain predefined functions when requested by a calling program unit.

Representative system service calls include a command to carry out a "read" function. A "read" request, based on supplied parameters, could request a read from a disk drive or other types of magnetic storage, or could request a read from a terminal or other devices.

Alternately, the operating system might support a system service call, such as a "write" to a storage unit or a device. A "write" request could send data or programs to communication lines, printers, or the like. A more extensive list of system service calls of a type supported by Tandem's GUARDIAN Operating System is attached hereto as Exhibit A.

In accordance with the present invention, there is interposed between the program unit 30 and the operating system 34 a functional layer 36 which includes the "user hooks" or entry points. At these points, an operator, a user, or a site can expand upon or modify external references or calls intercepted by the operating system.

Once an instruction has been intercepted, a first user hook is then checked or executed. This entry point can include an initial block of user or operator supplied code. This initial or "prior" block is to be executed before any modification and/or enhancement of

the function which is the subject of the intercepted instruction is carried out.

The intercepted call or service request may then be executed as required. This execution, as described below, can be modified and/or enhanced, or expanded upon in a predetermined fashion.

Then, a second user hook or entry point may be checked or executed to determine whether or not there is any post-translation, user, or site specific code which is to be executed. If so, that code is executed. Finally, appropriate parameters and/or data may be returned to the program unit 30 which had previously made the service request or call.

In accordance with the present invention, the interception process is carried out in one embodiment using a hierarchy that is very often imposed by the operating system between program library calls and system library calls. As a first step in carrying out the call or the functional request, if a program library 30a is associated with the program unit 30, the operating system 34 checks the program library 30a first to determine if the intercepted external reference or call is present in the program library.

By providing counterparts in the library 30a to some or all of the system service calls or functions of the operating system 34 before the operating system intercepts requests for such services from the program unit, the corresponding procedure in the program (not the system) library will be executed. This provides a vehicle to modify or expand such requests in a pre-determined fashion.

Hence, by associating with the program library structure 3oa, a plurality of modified operating system calls, when the program 30 executes a particular service call, service can be provided in accordance with that

request. In addition, on a substantially transparent basis to the executing program unit, the service can be enhanced and/or modified, or completely changed in a predetermined fashion. If and when the appropriate parameters and/or data are then returned to the program unit 30, that program can then continue executing subsequent instructions.

It will be understood that the library 30a is not required to practice the present method. An equivalent structure can be implemented in the operating system 34 as discussed subsequently or in the program unit 30 itself.

Example 1 illustrates the process. Subsequently referred to line numbers are listed along the left-hand margin of Example 1.

In Example 1, a read operation present in the program unit 30 could be intercepted and/or modified or translated on a substantially transparent basis in the interface layer 36. Line 40 of Example 1, defines the procedure to be executed as a "read" function with n parameters associated therewith.

The read process begins in a line 42. Line 44 represents a first user hook or entry point. A call is made to a procedure which includes one or more previously specified site specific or operator specific instructions which are not normally part of the "read" procedure. Subsequent to the execution of the procedure of line 44, the actual "read" procedure can be carried out as indicated schematically in line 46.

It should be noted that the actual read procedure which could be carried out could be a read procedure which is expanded and/or substantially different from the originally contemplated and specified read procedure in the calling program unit 30. Thus, a

bridging function can be provided, if necessary, between different program versions and/or releases.

```
40      PROC READ (1, 2 .... n)
42      BEGIN
44      CALL PRIOR (1, 2 .... n)
46      JUMP TO READ FUNCTION VIA O/S LOGICAL ADDRESS
48      CALL POST (1, 2 .... n)
50      END
        PROC PRIOR (1, 2 .... n)
        BEGIN           USER INSTRUCTIONS CAN BE
                        INSERTED AT THIS POINT IF
                        DESIRED
        END

        PROC POST (1, 2 .... n)

        BEGIN           USER INSTRUCTIONS CAN BE
                        INSERTED AT THIS POINT IF
                        DESIRED
        END
```

EXAMPLE 1

Line 48 is a second user hook or entry point. A procedure is called which includes one or more site specific or operator specific instructions which may be carried out after the read function is carried out. The end of the procedure is indicated in line 50.

It will be understood that the location, number, or function of the user hooks are not a limitation of the present invention. In addition, the present invention contemplates the use of multi-levels of entry points, such as in the program unit, the program library, or the system library.

Upon a return from the read procedure of Example 1 to the program unit 30, that program will continue execution which can be based on returned parameters or data, if any, which resulted from the read procedure initiated therein. Hence, information actually supplied to the program unit 30 could come from a completely different location and/or source than that originally contemplated by the program unit 30 and this change could be completely transparent thereto.

Figure 3 illustrates a flow diagram of an embodiment of the method of the present invention. The process of Figure 3 will be explained below in combination with the text of Example 1. In the embodiment of Figure 3, the program library 30a has been previously linked to the program unit 30 and is available at run time. Using the above-noted hierarchal approach, the operating system 34 checks the library 30a first when the program unit 30 calls an external function or service, or tries to initiate execution of an external procedure.

The library 30a has been previously loaded with procedures corresponding to at least some of the external references for the program 30. The names of some of the previously loaded library procedures must be

the same as the names of system service calls that are to be expanded upon and/or modified. (Usually, this is regarded as an error to be carefully avoided!)

In addition, it is necessary to be able to acquire, usually via the operating system, the logical address(es) of the respective system service call(s) in the operating system's library to be intercepted. The respective library procedure requires this information to be able to call that service function without using the name thereof.

For instance, in Example 1, a "read" system service call is to be intercepted and/or modified. The program library, as a result, includes a PROC READ. In line 46, to call the actual read in the operating system library, a: JUMP TO LOGICAL ADDRESS OF SSC READ must be executed to prevent PROC READ from calling itself.

Referring to Figure 3, the execution of the program unit 30 has been previously initiated. Step 62 represents execution of the program unit 30 until an external request of some sort is made or until the program unit 30 is completed, at which point it terminates in a step 64.

In the event that the program unit 30 makes an external request, such as a request for a "read" or "write" for example, the operating system 34, in step 68, first checks the program library 30a, if any, to determine whether or not this function or procedure is found therein. If the called function, procedure, or external reference is located by the operating system 34 in the library 30a, for example, the "read" procedure of Example 1, that procedure is initiated.

In a step 72, the first user hook or entry point is encountered. This corresponds to the call at line 44 of Example 1. If there exists operator or site specific procedures and/or code, such steps should be

executed. This corresponds to carrying out the procedure of line 44 of Example 1.

In a step 76, the system service call or other function, called by program unit 30, is carried out, corresponding to carrying out the "read" function of line 46 of Example 1. The executed procedure from the operating system that is executed may be <u>different</u> from that contemplated by the creator of the program unit 30.

In a step 78, the second or "post" user hook or entry point is encountered. This corresponds to carrying out the procedure of line 48 of Example 1. Then, there is a return to execution of the program unit 30 in a step 80. While executing user hook instructions, alternate return paths, such as step 80a or step 80b could be provided by the user.

In this example, if the called procedure or service request is not found in the library 30a, and if it is in the system library, then, in a step 70, the requested service or procedure is carried out, perhaps in combination with other services of the operating system 34. Any necessary parameters and/or data are returned to the program unit 30 which continues executing in step 62a.

As can be observed from the process of Figure 3, as a result of the site specific user supplied pre-translation and/or pre-modification steps, the first user hook, such as the process 44, along with the post-translation or post-modification steps, such as the process 48, it is relatively easy for an operator and/or a user to provide extensions, translations, and/or modifications to the original function being requested by the program unit 30. These are all outside of the program unit 30 and are substantially transparent to it.

Figure 4 illustrates an alternate embodiment of the present invention. In the embodiment of Figure

4, the program unit 30 need not have a library 30a associated therewith.

However, the names of the procedures or system service calls in the operating system library have been previously altered to distinguish them from the called procedure or "system service call" to be intercepted. With this change, the actual operating system call, under the new name, can subsequently be made. One of these procedures could correspond to the "open" procedure. Renaming pertinent system service routines in the system library, such as "open to "sopen", as illustrated in Figure 4, step 34b, can be done when the operating system is compiled and linked together. In addition, corresponding procedures, as illustrated in Figure 4, step 72a, must be loaded into the system library with the original names of the system service calls to be intercepted.

If the respective system library procedures of the operating system had been previously modified and expanded upon as described above, it would be possible to carry out a corresponding user specified "prior" procedure as identified on line 44 of Example 1 in step 72a, analogous to the step 72 previously discussed. After executing corresponding and/or similar system service calls in step 76a, the user defined instructions represented by the "post" procedure of Example 1 can be executed in a plurality of steps 78a. Subsequently, the operating system 34 returns appropriate parameters and/or data, if any, to the program unit 30, which then continues executing in a step 62a.

Using the previously described method, either the embodiment of Figure 3 or that of Figure 4, makes it possible for a user and/or operator to upgrade, maintain, and/or modify program units, such as the unit 30, to deal with both a changing environment and also

changing functional requirements, now and in the future. It is also possible to modify and/or upgrade system service calls so as to provide substantially different and/or enhanced functions not previously available to the corresponding program units, such as the program unit 30, as well as operating system 34.

The above-described instruction interceptions are carried out at run-time, and are substantially transparent to the executing program unit. Source code for the program unit is not required to practice the present method.

By making the "user hooks" or entry points available, as described above, both before and after executing the corresponding system service calls, for example, users and/or operators will be able to more effectively manage, maintain, and upgrade their program units in a very cost effective fashion. Further, because the present method is substantially external to the respective program unit, there should be no impact to third party vendor or maintenance relationships.

- Additional representative examples of ways in which the methods of Figures 3 and/or 4 could be used include improved resource allocation in a multi-processor environment by including provision for user specific and/or operator specific modification to resource allocation routines. Redundant write operations can be provided when carrying out the write function to provide multiple, substantially transparent, sets of data which can be used for verification, disaster recovery functions or the like.

Thus, in accordance with the present invention a user interface is provided to, on a substantially transparent basis, modify requests made by an executing program unit for a variety of purposes. This modification process takes place substantially outside of the program unit. It can be substantially outside of the associated

operating system but can be readily modified by the operator and/or the user for purposes of customization.

The present invention has been discussed in terms of translating and/or modifying instructions at run time in a program unit, such as the exemplary program unit 30. It will be understood that the present methods can be used with any type of program unit, such as an application, a utility, or the like. Hence, the present method could also be used to translate and/or modify instructions in programs that may be routinely thought of as part of the operating system.

It will also be understood that the embodiments of Figures 3 and/or 4 could be combined. In addition, it is also within the spirit and scope of the present invention to alternately merge some of the procedures of the program library with the associated main program unit.

Example 2 is a further illustration of the method hereof in source code form.

From the foregoing, it will be observed that numerous variations and modifications may be effected without departing from the spirit and scope of the invention. It is to be understood that no limitation with respect to the specific apparatus illustrated herein is intended or should be inferred. It is, of course, intended to cover by the appended claims all such modifications as fall within the scope of the claims.

EXHIBIT A

PARTIAL LIST OF TANDEM'S GUARDIAN
OPERATING SYSTEM CALLS
(WITHOUT PARAMETERS)

ALTER

ALTER PRIORITY

APS DATA GETPARAM

CONTROL

CREATE

DEFINEADO

DEFINEINFO

MEASURINFO

NEWPROCESS

OPEN FILE

PRINTINFO

PRINTREAD

READ

WRITE

```
 1.  000000 0  0  ?INSPECT,SYMBOLS,SAVEABEND
 3.  000000 0  0  !{*****************************************************************
 4.  000000 0  0  !{                                                                }
 5.  000000 0  0  !{              Copyright 1992, By Overlord Inc., All rights reserved
 6.  000000 0  0  !{                              Author : Don Kennedy              }
 7.  000000 0  0  !{                                                                }
 8.  000000 0  0  !{ Purpose : To demonstrate a method of intercepting pre-exsisting computer
 9.  000000 0  0  !{           instructions in order to modify and or enhance pre-exsisting
10.  000000 0  0  !{           computer instructions and supply user hooks's, without the
11.  000000 0  0  !{           requirement and or need of one or more of the following :
12.  000000 0  0  !{                                                                }
13.  000000 0  0  !{             A. Viewing the original program unit(s) source code.
14.  000000 0  0  !{             B. Viewing the original program unit(s) object code.
15.  000000 0  0  !{             C. Re-compiling the original program unit(s) source code.
16.  000000 0  0  !{             D. Prior knowledge of the pre-exsisting computer instructions.
17.  000000 0  0  !{             E. A total knowledge of the users present and or future needs.
18.  000000 0  0  !{             F. A complete understanding of the pre-exsisting instructions.
19.  000000 0  0  !{             G. Knowing the users purpose or reasons for the interception.
20.  000000 0  0  !{             H. That all the pre-exsisting instructions be executed.
21.  000000 0  0  !{             I. All and or any of the user hooks be executed.
22.  000000 0  0  !{             J. The result of the pre-exsisting instructions change.
23.  000000 0  0  !{             K. Guidance or expertise from original authors or inventors.
24.  000000 0  0  !{             L. Any additional physical, electronic or mechanical device.
25.  000000 0  0  !{                                                                }
26.  000000 0  0  !{ NOTE : The procedure chosen to intercept is only used as an example and
27.  000000 0  0  !{         can and could be changed to or combined with any EXTERNAL procedure
28.  000000 0  0  !{         CALL from a program unit.
29.  000000 0  0  !{                                                                }
30.  000000 0  0  !{         The computer language chosen to demonstrate this example is only
31.  000000 0  0  !{         used as an example and can and could be changed to or combined with
32.  000000 0  0  !{         another computer language.
33.  000000 0  0  !{                                                                }
34.  000000 0  0  !{         The computer system chosen to demonstrate this example on is only
35.  000000 0  0  !{         used as an example and can and could be changed to or combined with
36.  000000 0  0  !{         another computer system or a number of combinations of computer
37.  000000 0  0  !{         systems.
38.  000000 0  0  !{                                                                }
39.  000000 0  0  !{         The intercept method used to demonstrate this example is only
40.  000000 0  0  !{         used as one example of the method and can and could be changed in
41.  000000 0  0  !{         order to comply with other computer systems operating systems.
42.  000000 0  0  !{                                                                }
43.  000000 0  0  !{*****************************************************************
44.  000000 0  0
```

EXAMPLE 2

```
46.  000000 0  0  !
47.  000000 0  0  !{*****************************************************************************}
48.  000000 0  0  !
49.  000000 0  0  !  Below are the Tandem Guardian C30.06 Operating System calls that MAY be
50.  000000 0  0  !  called by this program.
51.  000000 0  0  !
52.  000000 0  0  !{*****************************************************************************}
53.  000000 0  0  !
54.  000000 0  0  ?SOURCE $SYSTEM.SYSTEM.EXTDECS0( ABEND,
55.  000000 0  0  ?                                 DEBUG,
56.  000000 0  0  ?                                 MYPID,
57.  000000 0  0  ?                                 MYSYSTEMNUMBER,
58.  000000 0  0  ?                                 PROGRAMFILENAME,
59.  000000 0  0  ?                                 RESERVELCBS,
60.  000000 0  0  ?                                 SYSTEMENTRYPOINTLABEL )
Source file:  [2]  $SYSTEM.SYSTEM.EXTDECS0  1991-10-04  21:22:43
10.  000000 0  0  ?SECTION ABEND
11.  000000 0  0  PROC ABEND (CRTPID, STOPBACKUP, PERROR, COMPLCODE, TERMINFO, SSID, LENGTH,
12.  000000 1  0                TEXT) EXTENSIBLE(1), CALLABLE;
13.  000000 1  0    INT    .CRTPID;          !IN OPTIONAL
14.  000000 1  0                             !processid to be abended
15.  000000 1  0    INT    STOPBACKUP;       !IN OPTIONAL
16.  000000 1  0                             !if true, abend backup
17.  000000 1  0    INT    .PERROR;          !OUT OPTIONAL
18.  000000 1  0                             !returned error value
19.  000000 1  0    INT    COMPLCODE;        !IN OPTIONAL
20.  000000 1  0                             !completion code
21.  000000 1  0    INT    TERMINFO;         !IN OPTIONAL
22.  000000 1  0                             !termination info
23.  000000 1  0    INT    .EXT SSID;        !IN OPTIONAL
24.  000000 1  0                             !subsystem ID
25.  000000 1  0    INT    LENGTH;           !IN OPTIONAL
26.  000000 1  0                             !length of text in bytes
27.  000000 1  0    STRING .EXT TEXT;        !IN OPTIONAL
28.  000000 1  0                             !text
29.  000000 1  0  EXTERNAL;

COMPLCODE                    Variable      INT           Direct        L-013
CRTPID                       Variable      INT           Indirect      L-016
LENGTH                       Variable      INT           Direct        L-007
PERROR                       Variable      INT           Indirect      L-014
SSID                         Variable      INT           EXT Pointer   L-011
STOPBACKUP                   Variable      INT           Direct        L-015
TERMINFO                     Variable      INT           Direct        L-012
TEXT                         Variable      STRING        EXT Pointer   L-006

30.  000000 0  0
1282.  000000 0  0  ?SECTION DEBUG
1283.  000000 0  0  PROC DEBUG  CALLABLE, RESIDENT;
1284.  000000 1  0  EXTERNAL;
1285.  000000 0  0
4140.  000000 0  0  ?SECTION MYPID
4141.  000000 0  0  INT PROC MYPID  CALLABLE, RESIDENT;
4142.  000000 1  0  EXTERNAL;
4143.  000000 0  0
4148.  000000 0  0  ?SECTION MYSYSTEMNUMBER
4149.  000000 0  0  INT PROC MYSYSTEMNUMBER  CALLABLE;
```

```
 4150.    000000 1  0   EXTERNAL;
 4151.    000000 0  0
 4631.    000000 0  0   ?SECTION PROGRAMFILENAME
 4632.    000000 0  0   PROC PROGRAMFILENAME (NAME) CALLABLE;
 4633.    000000 1  0     INT    .NAME;          !OUT
 4634.    000000 1  0                            !FILE NAME
 4635.    000000 1  0   EXTERNAL;

NAME                            Variable        INT             Indirect        L-003

 4636.    000000 0  0
 4838.    000000 0  0   ?SECTION RESERVELCBS
 4839.    000000 0  0   PROC RESERVELCBS (RECEIVECNT, SENDCNT) CALLABLE;
 4840.    000000 1  0     INT    RECEIVECNT;     !IN
 4841.    000000 1  0                            !# TO RESERVE FOR $RECEIVE
 4842.    000000 1  0     INT    SENDCNT;        !IN
 4843.    000000 1  0                            !# TO RESERVE FOR SENDING
 4844.    000000 1  0   EXTERNAL;

RECEIVECNT                      Variable        INT             Direct          L-004
SENDCNT                         Variable        INT             Direct          L-003

 4845.    000000 0  0
 5971.    000000 0  0   ?SECTION SYSTEMENTRYPOINTLABEL
 5972.    000000 0  0   INT PROC SYSTEMENTRYPOINTLABEL (NAME, LEN) CALLABLE;
 5973.    000000 1  0     STRING .NAME;          !IN
 5974.    000000 1  0     INT    LEN;            !IN
 5975.    000000 1  0   EXTERNAL;

LEN                             Variable        INT             Direct          L-003
NAME                            Variable        STRING          Indirect        L-004

 5976.    000000 0  0
Source file:  [1]  $SYSTEM.SAVE.EXAMPLE  1992-12-04  12:13:07
   61.    000000 0  0   !
   62.    000000 0  0   ?SOURCE $SYSTEM.ZGUARD.PCPUCTL(  GETPEEKCONFIGURATION,
   63.    000000 0  0   ?                               GETPEEKSTATISTICS      )
Source file:  [3]  $SYSTEM.ZGUARD.PCPUCTL  1991-08-06  09:17:53
  187.    000000 0  0   ?SECTION GETPEEKSTATISTICS
  188.    000000 0  0   INT PROC  GETPEEKSTATISTICS( CPU, BUF, LEN, TIME, PIN, SYSTEM )
  189.    000000 1  0                                                    EXTENSIBLE, CALLABLE;
  190.    000000 1  0   INT          CPU,
  191.    000000 1  0          .EXT  BUF,
  192.    000000 1  0                LEN;
  193.    000000 1  0   FIXED       .TIME;
  194.    000000 1  0   INT         .PIN;
  195.    000000 1  0   INT          SYSTEM;
  196.    000000 1  0   EXTERNAL;

BUF                             Variable        INT             EXT Pointer     L-012
CPU                             Variable        INT             Direct          L-013
LEN                             Variable        INT             Direct          L-010
PIN                             Variable        INT             Indirect        L-006
SYSTEM                          Variable        INT             Direct          L-005
TIME                            Variable        FIXED (0)       Indirect        L-007

  227.    000000 0  0   ?SECTION GETPEEKCONFIGURATION
```

SUBSTITUTE SHEET (RULE 26)

```
 228.    000000 0  0  INT PROC  GETPEEKCONFIGURATION( CPU, BUF, LEN, TIME, PIN, SYSTEM )
 229.    000000 1  0                                                    EXTENSIBLE, CALLABLE;
 230.    000000 1  0  INT          CPU,
 231.    000000 1  0          .EXT  BUF,
 232.    000000 1  0                LEN;
 233.    000000 1  0  FIXED        .TIME;
 234.    000000 1  0  INT          .PIN;
 235.    000000 1  0  INT           SYSTEM;
 236.    000000 1  0  EXTERNAL;
```

BUF	Variable	INT	EXT Pointer	L-012
CPU	Variable	INT	Direct	L-013
LEN	Variable	INT	Direct	L-010
PIN	Variable	INT	Indirect	L-006
SYSTEM	Variable	INT	Direct	L-005
TIME	Variable	FIXED (0)	Indirect	L-007

```
Source file:  [1]  $SYSTEM.SAVE.EXAMPLE  1992-12-04  12:13:07
   64.    000000 0  0  !
```

```
    66.     000000 0  0   !
    67.     000000 0  0   !{*****************************************************************}
    68.     000000 0  0   !{                                                                 }
    69.     000000 0  0   !{ Below are the pre and post EXTERNAL user hook definitions for the }
    70.     000000 0  0   !{ Tandem Guardian TOSVERSION intercept procedures.                 }
    71.     000000 0  0   !{                                                                 }
    72.     000000 0  0   !{ NOTE : The names chosen for the pre and post user hooks are only used as }
    73.     000000 0  0   !{        an example and these procedures can and could be named other names. }
    74.     000000 0  0   !{                                                                 }
    75.     000000 0  0   !{        The parameters chosen to be passed to the pre and post user hooks }
    76.     000000 0  0   !{        are used only for example and the number of paramaters and or names }
    77.     000000 0  0   !{        can and could be changed, deleted, and or added to.      }
    78.     000000 0  0   !{                                                                 }
    79.     000000 0  0   !{        In this example the pre user hook procedure "before^TOSVERSION^CALL" }
    80.     000000 0  0   !{        is called from the intercept TOSVERSION procedure passing the Tandem }
    81.     000000 0  0   !{        Guardian procedure TOSVERSION address.                   }
    82.     000000 0  0   !{                                                                 }
    83.     000000 0  0   !{        In this example the post user hook procedure "after^TOSVERSION^CALL" }
    84.     000000 0  0   !{        is called from the intercept TOSVERSION procedure passing the }
    85.     000000 0  0   !{        Tandem Guardian version level received from the Tandem Guardian }
    86.     000000 0  0   !{        procedure TOSVERSION.                                    }
    87.     000000 0  0   !{                                                                 }
    88.     000000 0  0   !{*****************************************************************}
    89.     000000 0  0   !
    90.     000000 0  0       PROC before^TOSVERSION^CALL( TOSVERSION^procedure^address );
    91.     000000 1  0         INT TOSVERSION^procedure^address;
    92.     000000 1  0       EXTERNAL;

TOSVERSION^PROCEDURE^ADDRESS     Variable        INT             Direct          L-003

    93.     000000 0  0   !
    94.     000000 0  0       PROC after^TOSVERSION^CALL( Guardian^version^level );
    95.     000000 1  0         INT Guardian^version^level;
    96.     000000 1  0       EXTERNAL;

GUARDIAN^VERSION^LEVEL           Variable        INT             Direct          L-003

    97.     000000 0  0   !
```

```
 99.  000000 0  0  !
100.  000000 0  0  !{*************************************************************************}
101.  000000 0  0  !{                                                                         }
102.  000000 0  0  !{ TOSVERSION: This is normally a Tandem Guardian operating system CALL that }
103.  000000 0  0  !{ program unit(s) would CALL to see what operating system the program unit(s) }
104.  000000 0  0  !{ are running on. Then the program unit(s) can determine if the version is a }
105.  000000 0  0  !{ proper version of the operating system, and proper action can be taken if }
106.  000000 0  0  !{ needed.                                                                  }
107.  000000 0  0  !{                                                                         }
108.  000000 0  0  !{ Since this program unit does not have a MAIN procedure it can be used as a }
109.  000000 0  0  !{ User and or System library program unit.                                 }
110.  000000 0  0  !{                                                                         }
111.  000000 0  0  !{ When program unit(s) use this program unit as a library, this program unit }
112.  000000 0  0  !{ will intercept the TOSVERSION CALL to the Tandem Guardian operating system }
113.  000000 0  0  !{ and user hooks CAN be utilized to modify and or enhance the original    }
114.  000000 0  0  !{ program unit(s).                                                        }
115.  000000 0  0  !{                                                                         }
116.  000000 0  0  !{ Since many program units may CALL this particular procedure as part of  }
117.  000000 0  0  !{ their initialization process, it is a good candidate to intercept, so that }
118.  000000 0  0  !{ the original program unit(s) can be enhanced and or modified.          }
119.  000000 0  0  !{                                                                         }
120.  000000 0  0  !{ In This example, TOSVERSION will be intercepted in order to allow user }
121.  000000 0  0  !{ hooks to be placed prior to the real TOSVERSION Tandem Guardian CALL as }
122.  000000 0  0  !{ well as after in order to modify or enhance the pre-exsisting program  }
123.  000000 0  0  !{ unit(s).                                                                }
124.  000000 0  0  !{                                                                         }
125.  000000 0  0  !{*************************************************************************}
126.  000000 0  0  !
```

```
129.    000000 0  0  !
129.    000000 0  0  INT PROC TOSVERSION;
130.    000000 1  0  !
131.    000000 1  0       BEGIN
132.    000000 1  1  !
133.    000000 1  1         INT     .Copyright[0:15]          := ["Copyright By Overlord Inc., 1992"],
134.    000020 1  1                 .Author[0:8]              := ["Donald J. Kennedy "],
135.    000031 1  1                 .overlord^version[0:14]   := ["G90C30.06.00.OLRDC30.06.A10.00"];
136.    000050 1  1  !
137.    000050 1  1         INT      G                         = 'G',
138.    000050 1  1                  L                         = 'L',
139.    000050 1  1                  S                         = 'S',
140.    000050 1  1                  E                        := 0;
141.    000050 1  1  !
142.    000050 1  1         INT      pre^user^hook^present    := @before^TOSVERSION^CALL,
143.    000050 1  1                  post^user^hook^present   := @after^TOSVERSION^CALL;
144.    000050 1  1  !
145.    000050 1  1         INT      version^level            :=  0,
146.    000050 1  1                  TOSVERSION^address       :=  0;
147.    000050 1  1  !
```

```
149.  000050 1  1  !
150.  000050 1  1  !{*****************************************************************}
151.  000050 1  1  !{                                                                 }
152.  000050 1  1  !{ NOTE: Using this program unit as a library program unit :      }
153.  000050 1  1  !{                                                                 }
154.  000050 1  1  !{       System Library;                                           }
155.  000050 1  1  !{                                                                 }
156.  000050 1  1  !{         If this program unit is going to be used as a System Library
157.  000050 1  1  !{         program unit then the Tandem Guardian TOSVERSION procedure name
158.  000050 1  1  !{         should be changed to another name using the BIND utility on the
159.  000050 1  1  !{         pre-exsisting System Library.                           }
160.  000050 1  1  !{                                                                 }
161.  000050 1  1  !{         The "TOSVERSION^name" variable should be changed to the new name,
162.  000050 1  1  !{         also the "TOSVERSION^name^length" variable should be changed to the
163.  000050 1  1  !{         length in BYTES of the new name.                        }
164.  000050 1  1  !{                                                                 }
165.  000050 1  1  !{         This program unit should then be compiled and BOUND with the proper
166.  000050 1  1  !{         pre-exsisting System Library program unit prior to the OSIMAGE
167.  000050 1  1  !{         being built.                                            }
168.  000050 1  1  !{                                                                 }
169.  000050 1  1  !{         WARNING : Do not CALL ABEND, DEBUG or STOP in any user hooks if
170.  000050 1  1  !{                   this program unit will be used as a System Library
171.  000050 1  1  !{                   program unit.                                 }
172.  000050 1  1  !{                                                                 }
173.  000050 1  1  !{       With pre-exsisting User Library;                          }
174.  000050 1  1  !{                                                                 }
175.  000050 1  1  !{         This program unit should be compiled and BOUND with the
176.  000050 1  1  !{         pre-exsisting User Library program unit and a new User library
177.  000050 1  1  !{         program unit should be created and linked to the proper program
178.  000050 1  1  !{         unit(s).                                                }
179.  000050 1  1  !{                                                                 }
180.  000050 1  1  !{       User Library;                                             }
181.  000050 1  1  !{                                                                 }
182.  000050 1  1  !{         This program unit should be compiled and then linked with the
183.  000050 1  1  !{         proper program unit(s).                                 }
184.  000050 1  1  !{                                                                 }
185.  000050 1  1  !{*****************************************************************}
186.  000050 1  1  !
```

```
188.    000050 1  1  !
189.    000050 1  1  !{*************************************************************************}
190.    000050 1  1  !{                                                                         }
191.    000050 1  1  !{ NOTE: Using this program unit as part of a MAIN program unit:          }
192.    000050 1  1  !{                                                                         }
193.    000050 1  1  !{       This program unit can be combined with a pre-exsisting MAIN program
194.    000050 1  1  !{       unit by doing the following :                                     }
195.    000050 1  1  !{                                                                         }
196.    000050 1  1  !{         A. BIND this program unit into a pre-exsisting MAIN program unit.
197.    000050 1  1  !{                                                                         }
198.    000050 1  1  !{*************************************************************************}
199.    000050 1  1  !
```

```
000050 1  1  !
000050 1  1          INT     .TOSVERSION^name[0:4]    :=  ["TOSVERSION"],
000055 1  1                   TOSVERSION^name^length := 10;
000055 1  1  !
000055 1  1          STRING .s^TOSVERSION^name       := @TOSVERSION^name '<<' 1;
000055 1  1  !
000055 1  1          TOSVERSION^address := SYSTEMENTRYPOINTLABEL( s^TOSVERSION^name,
000127 1  1                                                       TOSVERSION^name^length );
000134 1  1  !
```

```
211.    000134 1  1   !
212.    000134 1  1   !|*****************************************************************|
213.    000134 1  1   !|                                                                 |
214.    000134 1  1   !| Check to see if the pre^user hook procedure is present and if so then CALL |
215.    000134 1  1   !| the before^TOSVERSION^CALL prior to executing the Tandem Guardian |
216.    000134 1  1   !| TOSVERSION CALL and pass the address of the TOSVERSION procedure. |
217.    000134 1  1   !|                                                                 |
218.    000134 1  1   !|*****************************************************************|
219.    000134 1  1   !
220.    000134 1  1           IF pre^user^hook^present > 0 THEN
221.    000137 1  1             CALL before^TOSVERSION^CALL( TOSVERSION^address );
222.    000142 1  1   !
```

```
224.    000142 1  1  |
225.    000142 1  1  |{-------------------------------------------------------------------------}
226.    000142 1  1  |{                                                                         }
227.    000142 1  1  |{ NOTE : It is possible that the before^TOSVERSION^CALL procedure could   }
228.    000142 1  1  |{         execute a RETURN statement with it's own values which would return}
229.    000142 1  1  |{         control to the originating program unit(s) and not execute the  }
230.    000142 1  1  |{         remaining intructions in this procedure.                        }
231.    000142 1  1  |{                                                                         }
232.    000142 1  1  |{         This could be of value to the user if some program unit(s) must }
233.    000142 1  1  |{         be told that they are on operating system versions that they are}
234.    000142 1  1  |{         not really on and do not require the real operating system version}
235.    000142 1  1  |{         to be checked.                                                  }
236.    000142 1  1  |{                                                                         }
237.    000142 1  1  |{-------------------------------------------------------------------------}
238.    000142 1  1  |
```

```
240.    000142 1  1   !
241.    000142 1  1   !|*****************************************************************|
242.    000142 1  1   !|                                                                 |
243.    000142 1  1   !| CALL the Tandem Guardian TOSVERSION procedure and get the Tandem Guardian |
244.    000142 1  1   !| operating system release level.                                  |
245.    000142 1  1   !|                                                                 |
246.    000142 1  1   !|*****************************************************************|
247.    000142 1  1   !
248.    000142 1  1   ?RP 0
249.    000142 1  1   !                              |
250.    000142 1  1           STACK TOSVERSION^address;
251.    000143 1  1           CODE( DPCL );
252.    000144 1  1           STORE version^level;
253.    000145 1  1   !
```

```
255.    000145 1  1   !
256.    000145 1  1   !{*************************************************************************}
257.    000145 1  1   !{                                                                         }
258.    000145 1  1   !{ Check to see if the post^user hook is present and if so then CALL the   }
259.    000145 1  1   !{ after^TOSVERSION^CALL prior to returing the value of the Tandem Guardian }
260.    000145 1  1   !{ Operating system level to the originating program unit(s).              }
261.    000145 1  1   !{                                                                         }
262.    000145 1  1   !{*************************************************************************}
263.    000145 1  1   !                          |
264.    000145 1  1           IF post^user^hook^present > 0 THEN
265.    000150 1  1             CALL after^TOSVERSION^CALL( version^level );
266.    000153 1  1   |
```

```
268.    000153 1  1   !
269.    000153 1  1   !{*************************************************************************}
270.    000153 1  1   !{                                                                         }
271.    000153 1  1   !{ NOTE : It is possible that the after^TOSVERSION^CALL procedure could   }
272.    000153 1  1   !{        execute a RETURN statement with it's own values which would return}
273.    000153 1  1   !{        control to the originating program unit(s) and not execute the   }
274.    000153 1  1   !{        remaining intructions in this procedure.                         }
275.    000153 1  !   !{                                                                         }
276.    000153 1  1   !{        This could be of value to the user if some program unit(s) must  }
277.    000153 1  1   !{        be told that they are on a operating system version that they are }
278.    000153 1  1   !{        not really on, but do require that the real operating system version}
279.    000153 1  1   !{        first be checked.                                                }
280.    000153 1  1   !{                                                                         }
281.    000153 1  1   !{*************************************************************************}
282.    000153 1  1   !
```

```
 284.    000153 1  1   !
 285.    000153 1  1   !{*****************************************************************}
 286.    000153 1  1   !{                                                                 }
 287.    000153 1  1   !{ RETURN the value from the Tandem Guardian TOSVERSION procedure which will }
 288.    000153 1  1   !{ return control back to the original program unit that made the CALL to the }
 289.    000153 1  1   !{ Tandem Guardian TOSVERSION procedure.                           }
 290.    000153 1  1   !{                                                                 }
 291.    000153 1  1   !{*****************************************************************}
 292.    000153 1  1   !
 293.    000153 1  1         RETURN version^level;
 294.    000155 1  1   !
 295.    000155 1  1       END;
```

```
AUTHOR                          Variable    INT      Indirect   L+002
COPYRIGHT                       Variable    INT      Indirect   L+001
E                               Variable    INT      Direct     L+004
G                               Variable    INT      Direct     G+000
L                               Variable    INT      Direct     L+000
OVERLORD^VERSION                Variable    INT      Indirect   L+003
POST^USER^HOOK^PRESENT          Variable    INT      Direct     L+006
PRE^USER^HOOK^PRESENT           Variable    INT      Direct     L+005
S                               Variable    INT      Direct     S-000
S^TOSVERSION^NAME               Variable    STRING   Indirect   L+013
TOSVERSION^ADDRESS              Variable    INT      Direct     L+010
TOSVERSION^NAME                 Variable    INT      Indirect   L+011
TOSVERSION^NAME^LENGTH          Variable    INT      Direct     L+012
VERSION^LEVEL                   Variable    INT      Direct     L+007
```

```
000000  041557 070171 071151 063550 072040 041171 020117 073145    000010  071154 067562 062040 044556 061456 026040 030471 034462
000020  042157 067141 066144 020112 027040 045545 067156 062544    000030  074440 043471 030103 031460 027060 033056 030060 027117
000040  046122 042103 031460 027060 033056 040461 030056 030060    000050  052117 051526 042522 051511 047516 070414 070434 070445
000060  100000 024733 010401 000000 020376 000454 024700 010401    000070  000000 020376 000454 000002 070464 100012 024744 170411
000100  030001 024700 002055 170401 000025 003673 100020 026047    000110  170402 000025 003706 100011 026047 170403 000025 003712
000120  100017 026047 170411 000025 003724 100005 026047 170413    000130  040412 024711 027000 044410 040405 001000 016003 040410
000140  024700 027000 040410 000032 044407 040406 001000 016003    000150  040407 024700 027000 040407 125003
```

```
 296.    000000 0  0   !
```

```
 298.    000000 0  0   !
 299.    000000 0  0   !{*****************************************************************}
 300.    000000 0  0   !{                                                                 }
 301.    000000 0  0   !{ before^TOSVERSION^CALL : This procedure ( IF PRESENT ) can intercept the
 302.    000000 0  0   !{ pre-exsisting CALL to TOSVERSION prior to the CALL being placed. Several
 303.    000000 0  0   !{ uses can be made of this user hook. The following list is just a sample
 304.    000000 0  0   !{ of one or more possible ways this user hook could be used:
 305.    000000 0  0   !{                                                                 }
 306.    000000 0  0   !{   A. Do not call the real TOSVERSION, instead RETURN a value from here.
 307.    000000 0  0   !{   B. Add additional computer instructions prior to the TOSVERSION CALL.
 308.    000000 0  0   !{   C. Log a message that the pre-exsisting instructions CALL TOSVERSION.
 309.    000000 0  0   !{   D. Log a message of the time pre-exsisting instructions CALL TOSVERSION.
 310.    000000 0  0   !{   E. Count the number of times that TOSVERSION is CALLED.
 311.    000000 0  0   !{   F. Calculate the time between CALLS to TOSVERSION.
 312.    000000 0  0   !{   G. Gain access to pre-exsisting data values in the program unit(s).
 313.    000000 0  0   !{   H. Change pre-exsisting data values in the program unit(s).
 314.    000000 0  0   !{                                                                 }
 315.    000000 0  0   !{ The user can change the scope of the original program unit(s) by using this
 316.    000000 0  0   !{ hook without one or more of the requirements listed earlier. This allows
 317.    000000 0  0   !{ the user to have additional options and control on the modification and or
 318.    000000 0  0   !{ enhancements of the pre-exsisting computer instructions without the need
 319.    000000 0  0   !{ of the original authors and or inventors guidance or expertise.
 320.    000000 0  0   !{                                                                 }
 321.    000000 0  0   !{*****************************************************************}
 322.    000000 0  0   !
 323.    000000 0  0   PROC before^TOSVERSION^CALL( TOSVERSION^procedure^address );
 324.    000000 1  0     INT TOSVERSION^procedure^address;
 325.    000000 1  0   !
 326.    000000 1  0   !{*****************************************************************}
 327.    000000 1  0   !{                                                                 }
 328.    000000 1  0   !{  NOTE : This procedure can be removed from this program unit if needed.
 329.    000000 1  0   !{                                                                 }
 330.    000000 1  0   !{         If left empty, as it is, it will still be called but no additional
 331.    000000 1  0   !{         logic will be executed.                                 }
 332.    000000 1  0   !{                                                                 }
 333.    000000 1  0   !{         User computer instructions can be placed in this area that
 334.    000000 1  0   !{         should be executed prior the the REAL TOSVERSION procedure CALL
 335.    000000 1  0   !{         being made if needed.                                   }
 336.    000000 1  0   !{                                                                 }
 337.    000000 1  0   !{*****************************************************************}
 338.    000000 1  0   !
 339.    000000 1  0     BEGIN
 340.    000000 1  1     END;

TOSVERSION^PROCEDURE^ADDRESS      Variable        INT             Direct          L-003

 76000  125004

 341.    000000 0  0   !
```

```
343.  000000 0  0  !
344.  000000 0  0  !{*****************************************************************************}
345.  000000 0  0  !{                                                                             }
346.  000000 0  0  !{ after^TOSVERSION^CALL : This procedure ( IF PRESENT ) can intercept the    }
347.  000000 0  0  !{ pre-exsisting CALL to TOSVERSION after the CALL was placed. Several        }
348.  000000 0  0  !{ uses can be made of this user hook. The following list is just a sample     }
349.  000000 0  0  !{ of one or more possible ways this user hook could be used:                  }
350.  000000 0  0  !{                                                                            }
351.  000000 0  0  !{   A. Do not RETURN the value from TOSVERSION, instead RETURN another value.}
352.  000000 0  0  !{   B. Add additional computer instructions after the TOSVERSION CALL.      }
353.  000000 0  0  !{   C. Log a message that the pre-exsisting instructions CALL TOSVERSION.  }
354.  000000 0  0  !{   D. Log a message of the times pre-exsisting instructions CALL TOSVERSION.}
355.  000000 0  0  !{   E. Count the times that TOSVERSION is CALLED.                          }
356.  000000 0  0  !{   F. Calculate the time it took to call TOSVERSION.                     }
357.  000000 0  0  !{   G. Gain access to pre-exsisting data values in the program unit(s).   }
358.  000000 0  0  !{   H. Change pre-exsisting data values in the program unit(s).          }
359.  000000 0  0  !{                                                                            }
360.  000000 0  0  !{ The user can change the scope of the original program unit by using this  }
361.  000000 0  0  !{ hook without one or more of the requirements listed earlier. This allows  }
362.  000000 0  0  !{ the user to have additional options and control on the modification and or  }
363.  000000 0  0  !{ enhancements of the pre-exsisting computer instructions without the need  }
364.  000000 0  0  !{ of the original authors and or inventors, guidance and or expertise.     }
365.  000000 0  0  !{                                                                            }
366.  000000 0  0  !{*****************************************************************************}
367.  000000 0  0  !
368.  000000 0  0  PROC after^TOSVERSION^CALL( Guardian^version^level );
369.  000000 1  0    INT Guardian^version^level;
370.  000000 1  0  ;
```

```
372.    000000 1  0   !
373.    000000 1  0   !{*****************************************************************}
374.    000000 1  0   !{                                                                 }
375.    000000 1  0   !{  NOTE : This procedure can be removed from this program unit if needed.
376.    000000 1  0   !{                                                                 }
377.    000000 1  0   !{         If left empty, it will still be called, but no additional
378.    000000 1  0   !{         logic will be executed.                                 }
379.    000000 1  0   !{                                                                 }
380.    000000 1  0   !{         User computer instructions can be placed in this area that
381.    000000 1  0   !{         should be executed after the the REAL TOSVERSION procedure CALL
382.    000000 1  0   !{         is made, if needed.                                     }
383.    000000 1  0   !{                                                                 }
384.    000000 1  0   !{*****************************************************************}
385.    000000 1  0   !
```

```
387.  000000 1  0  !
388.  000000 1  0  !{*****************************************************************}
389.  000000 1  0  !{                                                                 }
390.  000000 1  0  !{ The following example is one of may uses that this user hook could be used }
391.  000000 1  0  !{ for.                                                            }
392.  000000 1  0  !{                                                                 }
393.  000000 1  0  !{ EXAMPLE : This hook will allow pre-exsisting program unit(s) using this }
394.  000000 1  0  !{            program unit as a library and making a CALL to TOSVERSION from }
395.  000000 1  0  !{            the original program unit(s), to also CALL the RESERVELCBS }
396.  000000 1  0  !{            procedure, before returning control back to the original program }
397.  000000 1  0  !{            unit(s).                                             }
398.  000000 1  0  !{                                                                 }
399.  000000 1  0  !{            Some of the pre-exsisting program units may CALL the Tandem }
400.  000000 1  0  !{            Guardian procedure RESERVELCBS later, in this case the later CALL }
401.  000000 1  0  !{            will override this CALL to RESERVELCBS and may require that the }
402.  000000 1  0  !{            RESERVELCBS be intercepted and similar user hook logic be placed }
403.  000000 1  0  !{            in the user hook "before^RESERVELCBS^CALL".           }
404.  000000 1  0  !{                                                                 }
405.  000000 1  0  !{            NOTE : The RESERVELCBS intercept procedure as well as the }
406.  000000 1  0  !{                   before^RESERVELCBS^CALL and after^RESERVELCBS^CALL are }
407.  000000 1  0  !{                   not included in this example but the intercept CONCEPT }
408.  000000 1  0  !{                   would be the same, and could also be incorporated in this }
409.  000000 1  0  !{                   program unit.                                 }
410.  000000 1  0  !{                                                                 }
411.  000000 1  0  !{            Other pre-exsisting program unit(s) may not currently CALL the }
412.  000000 1  0  !{            RESERVELCBS procedure. This user hook will allow these original }
413.  000000 1  0  !{            pre-exsisting program unit(s) to be modified and enhanced to CALL }
414.  000000 1  0  !{            the RESERVELCBS procedure if the proper system resources are }
415.  000000 1  0  !{            available and these pre-exsisting program unit(s) make a CALL }
416.  000000 1  0  !{            to the Tandem Guardian TOSVERSION procedure.         }
417.  000000 1  0  !{                                                                 }
418.  000000 1  0  !{*****************************************************************}
419.  000000 1  0  !
```

```
421.    000000 1  0  !
422.    000000 1  0    BEGIN
423.    000000 1  1  !
424.    000000 1  1      INT     .Copyright[0:15]          := ["Copyright 1992, By Overlord Inc."],
425.    000020 1  1              .Author[0:8]              := ["Donald J. Kennedy "],
426.    000031 1  1              .overlord^version[0:14]   := ["G90C30.06.00.OLRDC30.06.A10.00"];
427.    000050 1  1  !
428.    000050 1  1      STRUCT cpu^config^values( * );
429.    000050 1  1        BEGIN
430.    000050 1  2          INT      processor^type;
431.    000050 1  2          INT      total^pcbs;
432.    000050 1  2          INT      memory^size;
433.    000050 1  2          INT      syspool^size;
434.    000050 1  2          INT      mappool^size;
435.    000050 1  2          INT      total^lcbs;
436.    000050 1  2          INT      total^tles;
437.    000050 1  2          INT      total^bpts;
438.    000050 1  2        END;
439.    000050 1  1  !
440.    000050 1  1      STRUCT cpu^current^values( * );
441.    000050 1  1        BEGIN
442.    000050 1  2          FIXED    delta^time^cpu^idle;
443.    000050 1  2          INT      items^on^ready^list;
444.    000050 1  2          FIXED    delta^time^ready^queued;
445.    000050 1  2          INT(32)  page^fault^count;
446.    000050 1  2          INT      items^queued^for^memory;
447.    000050 1  2          FIXED    delta^time^memory^queued;
448.    000050 1  2          INT(32)  dispatch^count;
449.    000050 1  2          FIXED    delta^time^send^busy;
450.    000050 1  2          INT(32)  cache^hit^count;
451.    000050 1  2          INT(32)  disc^io^count;
452.    000050 1  2          INT      pcb^free^count;
453.    000050 1  2          INT      memory^locked;
454.    000050 1  2          INT      current^syspool;
455.    000050 1  2          INT      current^mappool;
456.    000050 1  2          INT      current^lcbs;
457.    000050 1  2          INT      current^tles;
458.    000050 1  2          INT      current^bpts;
459.    000050 1  2        END;
460.    000050 1  1  !
```

```
462.  000050 1  1  !
463.  000050 1  1
464.  000050 1  1       INT .cpu^config^buffer [0:( $LEN( cpu^config^values  ) - 1 ) / 2 ],
465.  000050 1  1           .cpu^current^buffer[0:( $LEN( cpu^current^values ) - 1 ) / 2 ],
466.  000050 1  1           .cpu^config ( cpu^config^values  )  = cpu^config^buffer,
467.  000050 1  1           .cpu^current( cpu^current^values )  = cpu^current^buffer,
468.  000050 1  1           .current^program^file^name;
469.  000050 1  1  !
470.  000050 1  1       INT .EXT x^cpu^config^buffer            := $XADR( cpu^config^buffer  ),
471.  000050 1  1           .EXT x^cpu^current^buffer           := $XADR( cpu^current^buffer );
472.  000050 1  1  !
473.  000050 1  1       INT  my^cpu                             := 0,
474.  000050 1  1            my^system^number                   := 0,
475.  000050 1  1            my^pin                             := 0,
476.  000050 1  1            my^pid                             := 0,
477.  000050 1  1            current^percent^lcbs^free          := 0,
478.  000050 1  1            allocate^send^lcbs                 := 0,
479.  000050 1  1            allocate^receive^lcbs              := 0,
480.  000050 1  1            program^loop^counter               := 0,
481.  000050 1  1            cpu^config^length                  := $LEN( cpu^config^values ),
482.  000050 1  1            cpu^current^length                 := $LEN( cpu^current^values ),
483.  000050 1  1           .my^program^file^name[0:11]         := [ 12 * [" "]];
484.  000064 1  1  !
```

```
!
!|.....................................................................|
!|                                                                     |
!|              The threshold and table settings are located here      |
!|                                                                     |
!| NOTE : To add more programs change "max^programs" to increase the size of |
!|        the program^file^name^table and add program names and set the SEND |
!|        and RECEIVE LCB values to the proper values for each program in the |
!|        "program^file^name^table". Leave unused entries as *UNUSED* so the |
!|        empty table space will not be checked.                       |
!|                                                                     |
!|.....................................................................|
!
    INT lcb^saftey^threshold  :=  40; ! At least 40% of the LCB's must be free
                                      ! in order to RESERVE the LCB's in the
                                      ! table. IF not, no LCB's will be reserved
                                      ! even if the file name is found.
!
    LITERAL max^programs        =  15, ! maximum programs that the table can hold
            entry^length        =   6, ! Size of each table entry
            table^size          =  max^programs * entry^length;
!
    INT program^file^name^table[0:table^size - 1] :=
!
!       Program     Reserve    Reserve     Program
!       File        SEND       RECEIVE     Entry
!       Name        LCB's      LCB's       Number
!       --------    --------   --------    --------
        ["BINSERV ", 2        , 2         , ! 1
         "COBOL   ", 2        , 2         , ! 2
         "COMINT  ", 1        , 1         , ! 3
         "EDIT    ", 1        , 2         , ! 4
         "ENFORM  ", 3        , 3         , ! 5
         "FUP     ", 2        , 2         , ! 6
         "INSPECT ", 1        , 1         , ! 7
         "OLORDCPU", 1        , 1         , ! 8
         "OLORDSYS", 1        , 1         , ! 9
         "OLORDMON", 1        , 1         , ! 10
         "PERUSE  ", 1        , 2         , ! 11
         "SYMSERV ", 2        , 2         , ! 12
         "TACL    ", 1        , 1         , ! 13
         "TAL     ", 2        , 2         , ! 14
         "*UNUSED*", 1        , 1        ];! 15
!
    FIXED my^time     := 0F;

    my^pid            := MYPID;
    my^pin            := my^pid.<8:15>;
    my^cpu.<8:15>     := my^pid.<0:7>;
    my^system^number  := MYSYSTEMNUMBER;
```

```
538.  000332 1  1  !
539.  000332 1  1  !{*************************************************************************}
540.  000332 1  1  !{                                                                         }
541.  000332 1  1  !{       Get current system status to see what percent of LCB's are free    }
542.  000332 1  1  !{                                                                         }
543.  000332 1  1  !{*************************************************************************}
544.  000332 1  1  !
545.  000332 1  1      CALL GETPEEKCONFIGURATION( my^cpu,
546.  000332 1  1                                 x^cpu^config^buffer,
547.  000332 1  1                                 cpu^config^length,
548.  000332 1  1                                 my^time,
549.  000332 1  1                                 my^pin,
550.  000332 1  1                                 my^system^number );
551.  000346 1  1  !
552.  000346 1  1      CALL GETPEEKSTATISTICS( my^cpu,
553.  000346 1  1                              x^cpu^current^buffer,
554.  000346 1  1                              cpu^current^length,
555.  000346 1  1                              my^time,
556.  000346 1  1                              my^pin,
557.  000346 1  1                              my^system^number );
558.  000362 1  1  !
```

```
                                                Copyright 1992, by Overlord Inc.
    current^percent^lcbs^free
      := $INTR( $DBL( cpu^current.current^lcbs ) * 100D
        / $DBL( cpu^config.total^lcbs ) );

    @current^program^file^name := @program^file^name^table;

    IF lcb^saftey^threshold < current^percent^lcbs^free THEN
      BEGIN
        CALL PROGRAMFILENAME( my^program^file^name );
          FOR program^loop^counter := 0 TO ( max^programs - 1 ) DO
            BEGIN
              IF current^program^file^name = my^program^file^name[8] FOR 4 THEN
                BEGIN
                  allocate^send^lcbs    := current^program^file^name[4];
                  allocate^receive^lcbs := current^program^file^name[5];
                  CALL RESERVELCBS( allocate^send^lcbs, allocate^receive^lcbs );
                END;
              @current^program^file^name
                := @current^program^file^name '+' entry^length;
              IF current^program^file^name = "*UNUSED*" THEN
                program^loop^counter := ( max^programs - 1 );
            END;
      END;

  END;
```

```
ALLOCATE^RECEIVE^LCBS          Variable        INT             Direct          L+021
ALLOCATE^SEND^LCBS             Variable        INT             Direct          L+020
AUTHOR                         Variable        INT             Indirect        L+002
COPYRIGHT                      Variable        INT             Indirect        L+001
CPU^CONFIG                     Variable,20     STRUCT-1        Indirect        L+004
CPU^CONFIG^BUFFER              Variable        INT             Indirect        L+004
CPU^CONFIG^LENGTH              Variable        INT             Direct          L+023
CPU^CONFIG^VALUES              Variable        TEMPLATE ,20
  1 PROCESSOR^TYPE                    0,2      INT             Direct
  1 TOTAL^PCBS                        2,2      INT             Direct
  1 MEMORY^SIZE                       4,2      INT             Direct
  1 SYSPOOL^SIZE                      6,2      INT             Direct
  1 MAPPOOL^SIZE                     10,2      INT             Direct
  1 TOTAL^LCBS                       12,2      INT             Direct
  1 TOTAL^TLES                       14,2      INT             Direct
  1 TOTAL^BPTS                       16,2      INT             Direct
CPU^CURRENT                    Variable,102    STRUCT-1        Indirect        L+005
CPU^CURRENT^BUFFER             Variable        INT             Indirect        L+005
CPU^CURRENT^LENGTH             Variable        INT             Direct          L+024
CPU^CURRENT^VALUES             Variable        TEMPLATE ,102
  1 DELTA^TIME^CPU^IDLE               0,10     FIXED (0)       Direct
  1 ITEMS^ON^READY^LIST              10,2      INT             Direct
  1 DELTA^TIME^READY^QUEUED          12,10     FIXED (0)       Direct
  1 PAGE^FAULT^COUNT                 22,4      INT (32)        Direct
  1 ITEMS^QUEUED^FOR^MEMORY          26,2      INT             Direct
  1 DELTA^TIME^MEMORY^QUEUED         30,10     FIXED (0)       Direct
  1 DISPATCH^COUNT                   40,4      INT (32)        Direct
  1 DELTA^TIME^SEND^BUSY             44,10     FIXED (0)       Direct
  1 CACHE^HIT^COUNT                  54,4      INT (32)        Direct
  1 DISC^IO^COUNT                    60,4      INT (32)        Direct
```

1 PCB^FREE^COUNT	64,2	INT	Direct	
1 MEMORY^LOCKED	66,2	INT	Direct	
1 CURRENT^SYSPOOL	70,2	INT	Direct	
1 CURRENT^MAPPOOL	72,2	INT	Direct	
1 CURRENT^LCBS	74,2	INT	Direct	
1 CURRENT^TLES	76,2	INT	Direct	
1 CURRENT^BPTS	100,2	INT	Direct	
CURRENT^PERCENT^LCBS^FREE	Variable	INT	Direct	L+017
CURRENT^PROGRAM^FILE^NAME	Variable	INT	Indirect	L+006
ENTRY^LENGTH	Literal	INT	%000006	
GUARDIAN^VERSION^LEVEL	Variable	INT	Direct	L-003
LCB^SAFTEY^THRESHOLD	Variable	INT	Direct	L+026
MAX^PROGRAMS	Literal	INT	%000017	
MY^CPU	Variable	INT	Direct	L+013
MY^PID	Variable	INT	Direct	L+016
MY^PIN	Variable	INT	Direct	L+015
MY^PROGRAM^FILE^NAME	Variable	INT	Indirect	L+025
MY^SYSTEM^NUMBER	Variable	INT	Direct	L+014
MY^TIME	Variable	FIXED (0)	Direct	L+161
OVERLORD^VERSION	Variable	INT	Indirect	L+003
PROGRAM^FILE^NAME^TABLE	Variable	INT	Direct	L+027
PROGRAM^LOOP^COUNTER	Variable	INT	Direct	L+022
TABLE^SIZE	Literal	INT	%000132	
X^CPU^CONFIG^BUFFER	Variable	INT	EXT Pointer	L+007
X^CPU^CURRENT^BUFFER	Variable	INT	EXT Pointer	L+011

```
000000  041557 070171 071151 063550 072040 030471 034462 026040
000020  042157 067141 066144 020112 027040 045545 067156 062544
000040  046122 042103 031460 027060 033056 040461 030056 030060
000060  020040 020040 020040 020040 041111 047123 042522 053040
000100  041517 046511 047124 020040 000001 000001 042504 044524
000120  000003 000003 043125 050040 020040 020040 000002 000002
000140  042103 050125 000001 000001 047514 047522 042123 054523
000160  050105 051125 051505 020040 000001 000002 051531 046523
000200  000001 000001 052101 046040 020040 020040 000002 000002
000220  070740 003204 024700 070740 003214 024700 070740 003232
000240  024711 100000 170405 130001 000002 000002 024755 000002
000260  002132 000002 000002 024733 002135 170401 000025 003511
000300  000025 003530 100017 026047 170425 000025 003542 100014
000320  040416 006377 044415 040416 030110 100377 070413 000367
000340  005376 024777 100771 024700 027000 000107 040413 060411
000360  027000 000107 103036 143405 000327 100000 100144 000222
000400  040426 040417 000215 013047 170425 024700 027000 100000
000420  143406 044420 103005 143406 044421 040420 040421 024711
000440  000200 100004 026247 015002 100016 044422 040422 104001

000010  041171 020117 073145 071154 067562 062040 044556 061456
000030  074440 043471 030103 031460 027060 033056 030060 027117
000050  020040 020040 020040 020040 020040 020040 020040 020040
000070  000002 000002 041517 041117 046040 020040 000002 000002
000110  020040 020040 000001 000002 042516 043117 051115 020040
000130  044516 051520 042503 052040 000001 000001 047514 047522
000150  000001 000001 047514 047522 042115 047516 000001 000001
000170  042522 053040 000002 000002 052101 041514 020040 020040
000210  025125 047125 051505 042052 000001 000001 070565 024700
000230  024700 070740 003241 024700 002001 100000 170404 130001
000250  000002 100020 100102 024755 070740 003262 100050 024711
000270  100020 026047 170402 000025 003524 100011 026047 170403
000310  026047 070427 000025 003551 100132 026047 027000 044416
000330  027000 044414 040413 060407 040423 070561 070415 040414
000350  040424 070561 070415 040414 005376 024777 100771 024700
000370  102005 142404 000327 000223 000110 044417 070427 044406
000410  010437 170406 103010 173425 100004 026207 015012 103004
000430  027000 170406 003006 044406 170406 000025 100016 004400
000450  034422 001016 016336 125004 025125 047125 051505 042052
```

SUBSTITUTE SHEET (RULE 26)

GLOBAL MAP

ABEND	Proc		External
AFTER^TOSVERSION^CALL	Proc		%000216
BEFORE^TOSVERSION^CALL	Proc		%000000
DEBUG	Proc		External
GETPEEKCONFIGURATION	Proc	INT	External
GETPEEKSTATISTICS	Proc	INT	External
MYPID	Proc	INT	External
MYSYSTEMNUMBER	Proc	INT	External
PROGRAMFILENAME	Proc		External
RESERVELCBS	Proc		External
SYSTEMENTRYPOINTLABEL	Proc	INT	External
TOSVERSION	Proc	INT	%000055

ENTRY POINT MAP BY NAME FOR FILE: \CLXA.$WORK.COE.cpatent2

SP	PEP	BASE	LIMIT	ENTRY	ATTRS	NAME	DATE	TIME	LANGUAGE	SOURCE FILE
00	004	000163	000642	000401		AFTER^TOSVERSION^CALL	04DEC92	12:28	TAL	$SYSTEM.SAVE.EXAMPLE
00	003	000162	000162	000162		BEFORE^TOSVERSION^CALL	04DEC92	12:28	TAL	$SYSTEM.SAVE.EXAMPLE
00	002	000005	000161	000062		TOSVERSION	04DEC92	12:28	TAL	$SYSTEM.SAVE.EXAMPLE

WHAT IS CLAIMED IS:

1. An apparatus for translating one or more steps of a pre-existing method for carrying out a predetermined function, wherein user defined steps can be incorporated therein, comprising:

circuitry for detecting a step from the pre-existing method which is a candidate for a translation; and

circuitry for determining if a previously defined, user supplied, pre-translation set of steps is to be executed before executing any predetermined translation steps, and in response to the determining steps, executing the set of pre-translation steps where indicated.

2. An apparatus according to claim 1 further including means for determining if a previously defined, user supplied, post-translation set of steps is to be executed after executing any predetermined translation steps, and in response thereto, executing the post steps where indicated.

3. A process of translating one or more steps of a pre-existing method for carrying out a predetermined function, wherein user defined steps can be incorporated therein, in accordance with the apparatus of claim 1, comprising:

detecting a step from the pre-existing method which is a candidate for a translation; and

determining if a previously defined, user supplied, pre-translation set of steps is to be executed before executing any predetermined translation steps, and in response to the determining step, executing the set of pre-translation steps where indicated.

4. The process of claim 3 further including the step of:

determining if a previously defined, user supplied, post-translation set of steps is to be executed after executing any predetermined translation steps, and in response thereto, executing the post steps where indicated.

5. A method of executing a predefined set of steps, including altering one or more of the steps in a predetermined fashion wherein user defined steps can be incorporated therein, in accordance with the apparatus of claim 1, comprising:

detecting a step which is a candidate for alteration;

executing the altering steps; and

determining if a previously defined, user supplied, post-alteration set of steps is to be executed after executing the set of post-alteration steps where indicated.

6. The method of claim 5 further including, after the detecting step, the step of:

determining if a previously defined, user supplied, pre-alteration set of steps is to be executed before executing any predetermined altering steps, and in response to the determining step, executing the set of pre-alteration steps where indicated.

7. A method of intercepting and modifying pre-existing instructions at run time in a computer program being executed in an apparatus as in claim 1, comprising:

intercepting a selected instruction and determining if it is a candidate for modification;

subsequent to the modifying step, evaluating if a previously defined, operator supplied, post-modification set of steps exists;

executing the operator supplied, post-modification set of steps as indicated; and

returning to the sequence of steps immediately after the detected step, thereby continuing the process.

determining if an alterable, previously defined, pre-modification set of instructions is to be executed, and in response thereto, executing the pre-modification set of instructions, if any; and

modifying or executing the intercepted instruction.

8. The method of claim 7 further including the step of:

determining if an alterable, previously defined, post-modification set of instructions is to be executed, and in response thereto, executing the post-modification set of instructions, if any.

9. A method of allocating resources within a multiple node, multiple processor system, wherein at least some of the nodes are spaced apart and are interconnected by communication links, wherein one or more of the processors includes an apparatus as in claim 1, the method comprising:

carrying out a sequence of steps in a predetermined process in a selected processor at one of the nodes;

detecting a step in the sequence which is to be carried out and which is a candidate for run-time modification;

intercepting the detected step and evaluating if a previously defined, operator supplied, pre-modification set of steps exists;

interrupting the sequence and executing the operator supplied pre-modification set of steps as indicated;

modifying the candidate step using a predetermined sequence of one or more predetermined modifier steps;